VC SUMMARY 2017

Simone Guggiari

SIMONE GUGGIARI
ETH ZÜRICH 2017
LEGI # 14-932-867

2a. IMAGE SEGMENTATION

segmentation = classification problem partition img

$I = \bigcup_{i=1}^{n} R_i, \quad R_i \cap R_j = \emptyset \ \forall i \neq j$ (n regions)

quality depends on application you need

Berkley segm. DB (thickness of lines)

Thresholding produces binary img

$$B = \begin{cases} 1 & \text{if } I(x,y) \geq t \\ 0 & \text{otherwise} \end{cases}$$

parameter T (trial & error / automatic methods)

Chromakeying (green screen) → control light

plain distance measure

$I_\alpha = |I - g| > T$ → $I_{comp} = I_\alpha \cdot I_A + (1 - I_\alpha) I_B$ (hard alpha mask)

T ≥ 0, g = (0, 255, 0)

⚠ variation not same in all channels

Normalization $I = R + G + B$

$$I_{norm} = (R/I, G/I, B/I)$$

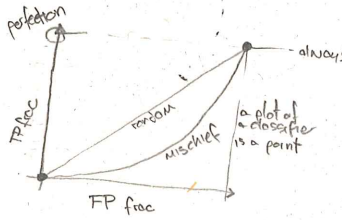
ROC analysis (receiver operating characteristics)
↳ characterize performance of binary classifier

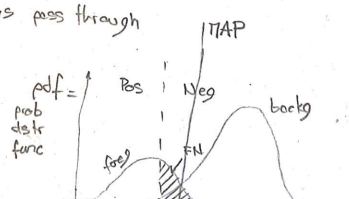
errors

	output V	output X
P = TP + FN		tot # real P
N = TN + FP		" N

TP fraction (sensitivity) $= \frac{TP}{P} = \frac{TP}{TP+FN}$

FP fraction (1 − specificity) $= \frac{FP}{N} = \frac{FP}{FP+TN}$





MAP (maximum a posteriori) detector

operating point → select one by assigning V/C: TP, TN, FP, FN (=0 (sometimes/simplicity)) value / cost

threshold to use: $\beta = \frac{N}{P} \frac{V_{TN} + C_{FP}}{V_{TP} + C_{FN}}$ (point on ROC with gradient)

context: gives extra info to better segment

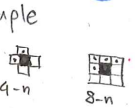
Performance assessment

• training set	tuning alg
• validation set	performance score
• test set	see in action

~~PIXEL CONNECTIVITY~~ Pixel connectivity

pixel = ~~square~~ sample

neighborhood



path between P_1, P_n is $\{P_1, P_2, \dots, P_n\}$ | P_i is 4/8-n of P_{i+1}

region connected (4/8-n way) if between any 2 pixels there is 4/8-n path

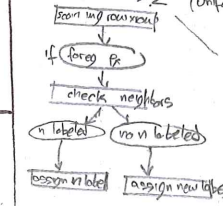
CCD (charged couple dev)
old / specific
costly / $ expens
blooming
sequential sensitive

CMOS
new
cheap
less sensitive
per px amplifier
smart px

rolling shutter → sequential readout of lines

bilinear interp

CCL (connected component labeling)
label each connected component of bin img with different #
→ only extract connected components of foreground
stage 1 → st2 (unify labels)



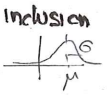
region growing (flooding) — union find DS (disjoint)
- start from point
- add neighb. satisfying criteria
- stop when done

see step 2

variations
seed region: • hand • automatic • multiple seeds (regions), photoshop

ALG CODE

inclusion criteria



include ⇔ $(I(x,y) - \mu)^2 < (n\sigma)^2$

snake active contour polygon (constraints), minimize $E = E_{tension} + E_{stiffness} + E_{img}$

2b. IS —

FG-BG segmentation
→ gaussian model per pixel

(I_μ, I_Σ) (μ ↓, Σ ↓)

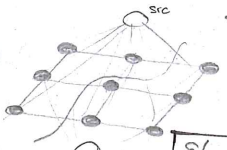
distance measures Mahalanobis dist $= (x-\mu)^T \Sigma^{-1} (x-\mu) > t$
estim from n data pts $\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_j - \bar{x})^T$

→ with $g = I_{bg}$ or better

$$I_\alpha = \sqrt{(I - I_{bg})^T \Sigma^{-1} (I - I_{bg})} > T \quad (\text{or } I < C)$$

↳ bg pixel appearance cov. matrix $\Sigma_g = E[(X_i - \mu_i)(X_j - \mu_j)]$ covariance

Adding spatial relations
Markov random fields (?)
(see markov chains)



edge = how much 2 px relate

use graph cut
→ minimize E cut = best segmentation = collection of edges (min cut)

cost to assign label to px

$$\mathcal{E}(c, \theta, data) = \sum_i \psi_1(c_i, \theta, data) + \sum_{i,j \in edges} \psi_2(c_i, c_j, \theta, data)$$

cost to assign pair of labels to connected px

user interaction
- point
- bbox
- strokes ← iterated
- in/out

iterated graph cuts
(more iterations, energy decreases)

border matting
- hard/soft segmentation → fit smoot alpha parameter with data

MORPHOLOGICAL OPERATORS

bool px transform (bin img)

ERODE (sn = minkowsky subtr) → erase if 1 of sn is bg → struct elem is fg
DILATE (sn = " add) → paint

goal: smooth & remove noise **E-E-D-D** = cleanup

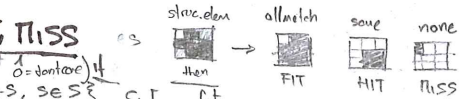
Structuring element (bin array)

img as set (the 1's) $I = \{(1,1), (2,1), \dots\}$

x = position
$s \in S$ struc.elem
(only considering the ones)

FIT, HIT, MISS
(only care about 1 = foreground)

struc.elem S: $\{q : q = x + s, s \in S\} \subset I$ → fit; $\cap I \neq \emptyset$ → hit; $\cap I = \emptyset$ → miss



$(S - I \text{ at } x)$ $+ \equiv 1$ $- \equiv 0$

Erosion ⊖
$E = I \ominus S$ (erode I with struc.elem S)

$f(x) = \begin{cases} 1 \\ 0 \end{cases}$ if S fits I at x

$E = \{x : x + s \in I \ \forall s \in S\}$

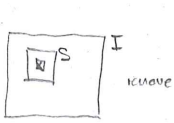
Dilation ⊕
$D = I \oplus S$ (dilate ...)

$D(x) = \begin{cases} 1 \\ 0 \end{cases}$ —— hits ——

$D = \{x : x - s, q \in I \wedge s \in S\}$

opening ○
$I \circ S = (I \ominus S) \oplus S$

closing ●
$I \bullet S = (I \oplus S) \ominus S$



remove isles in bg

(D) → { }

remove holes in fg

Morphological filtering
(remove holes & islands) using opening & closing
→ str.el shape determines result (circular is good)

Granulometry
size distrib of distinct regions in img → for increasing size for str.el & count regions

Hit & Miss
(core of bkgd, 180° x = don't care) ⊗ $H = I \otimes S$ (template matching) corner detection

1 if exactly the same as pos
0 otherwise

Thinning
$I \oslash S = I \setminus (I \otimes S)$

Thickening
$I \odot S = I \cup (I \otimes S)$

$(I \odot S)^c = I^c \oslash S$

Sequential: $I \oslash \{S_{i=1\dots n}\} = ((I \oslash S_1) \oslash \dots) \oslash S_n$ (set of single str.el = Golay alphabet)

Skeleton
(MAT = medial axis transform)
stick figure representation

B =

$S_n(X) = (X \ominus_n B) \setminus ((X \ominus_n B) \circ B)$ nth skeleton subset

$S(X) = \bigcup_{i=1}^{\infty} S_i(X)$

$X = \bigcup_{n=0}^{\infty} S_n(X) \oplus_n B$

$\ominus_n$ = n successive erosions

used for object/character recognition

grassfire on border where they meet

or union of centers of discs that touch 2 borders

3a. CONVOLUTION & FILTERING

Filtering
→ modify px on local fct on neigh

$L[\alpha I_1 + \beta I_2] = \alpha L[I_1] + \beta L[I_2]$

- local
- linear
- shift-invar (same everywh)

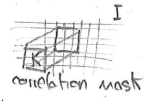
$F[x(t+a)] = q(t+a)$ not dep on time/position

weighted sum: $I'_j = \sum_{i=1}^{N} \alpha_{ij} I_i$ ($N = w \times h$) α_{ij} = elems of kernel ($\neq 0$ for [ ])

Correlation
$I' = K \circ I$ (templ. match)

$I'(x,y) = \sum_{(i,j) \in N(x,y)} K(i,j) I(x+i, y+j) = \sum_{i=-k}^{k} \sum_{j=-k}^{k}$

dot prod



correlation mask

high corr = high value = bright

index

K

Img

Convolution
(point spread)

$I' = K * I$

$I'(x,y) = \sum_{(i,j) \in N(x,y)} K(i,j) I(x-i, y-j)$ (correlation with reversal kernel)

$= \sum K(-i,-j) I(x+i, y+j)$

$\sum_{j=-k}^{k} \sum_{i=-k}^{k}$

$K(i,j) = K(-i,-j)$ corr = conv

one dual of other

properties conv:
- $f*g = g*f$
- ass. $(f*g)*h = f*(g*h)$
- distrib. $f*(g+h) = f*g + f*h$
- flip kernel

conv

identities

shift left

brightens

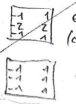
box filter

or

weighted smoothing filters

sum = 1

sharpen

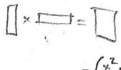


edge detector (cancels out when similar)

low pass, smooth

Separable kernel
$K(m,n) = f(m)\,g(n)$

kernels with rank 1



Gaussian
$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$

↳ separable! $\frac{1}{\sqrt{2\pi}\sigma} e^{\left(\frac{-x^2}{2\sigma^2}\right)} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{\left(\frac{-y^2}{2\sigma^2}\right)} = g(x)\,g(y)$

σ = spread of gaussian

Noise edges
- clip filter / vary filter
- wrap around
- copy edge / reflect
- ignore / copy / truncate
- pad with 0 / copy

Sobel
(2 passes (90°))

Prewitt
Roberts
discrete approx of 1st derivative
(detect change)
(diagonal gradients)

↔ differential filters

$I''(m,n) = \sum_{j=-N}^{N} g(j)\, I(m, n-j)$

$I'(m,n) = \sum_{i=-M}^{M} f(i)\, I''(m-i, n)$

Gaussian
weight contribution by nearness

how much scale varies

- look for strong responses in DoG over scale space
- only consider local maxima (both pos & scale space)

ellipses (EV) rotate, shapes (EW) don't change, rotations C not invariant

High pass filters = sharpen
take out image but highlight edges

Laplacian (2nd derivative) smoothing in related

high pass f. edge detect

Differentiation &
conv

$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$ FD

- linear
- shift inv

] convolution

conv

Sharpening (enhancement)
increase high-freq components

$I' = I + \alpha |K * I|$ ↳ high-pass

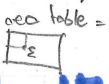
[0..1]

Integral images
efficiently compute conv using const. rectangle

es. efficient face detection

also possible along diagonal

summed area table = ds & alg to quickly generate sum of values in rectangular subset of grid



$\Sigma = D - C - B + A$

3b. IMAGE FEATURES

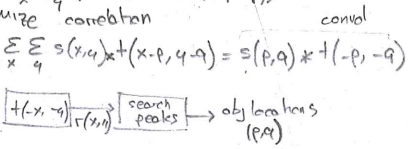
Template matching
locate template $t(x,y)$ in image $s(x,y)$

→ best match = minimize mean-sqr error

$\mathcal{E}(p,q) = \sum_x \sum_y (s(x,y) - t(x-p, y-q))^2$

→ maximize correlation

$r(p,q) = \sum_x \sum_y s(x,y)\, t(x-p, y-q) = s(p,q) * t(-p,-q)$ conv



$s(x,y) \to t(-x,-y) \to r(x,y) \to$ search peaks → obj locations (p,q)

Gradient:
isotropic (laplace isn't) = uniformity in all dirs

& less noisy (↳ 2nd der)

Gradient vs. Laplacian

Edge detection
$|\text{grad}(f(x,y))| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

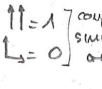
Digital img: finite differences

central diff [-1 0 1] diff [-1 1]

log sum of squared (diff) gradients with thresholds (Gillsface)

Laplacian operator
detect discontinuities using 2nd derivative $\nabla f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

↳ approx with 3×3 convolution (high pass filters)



f(x) edge, f' local maxima, f'' (detect zero crossing) → sensitive: blur first

L∘G (log of gauss)
$\text{LoG}(x,y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$

(precompute combined kernel)

this causes distortion

Canny edge detector
1) smooth (gauss) filter
2) comp gradient magn and angle
3) apply nonmaxima suppression (edge thinning) to gradient magn.
4) double threshold (strong/weak edges)
5) reject weak, unconnected with strong edge px (called hysteresis)

$|\nabla I| = \dots (D_x * I)$ discretize/given fct

$\Phi(x,y) = \text{grad}$ $\alpha(x,y) = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

weak, strong, low, high

$\Phi(\nabla I)$

edge px directions

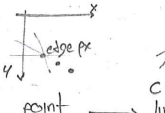
$M(x,y)$ smaller than neighbors in direction → suppress

Hough transform
fit straight line/curve to set of edge px (template matching)

filters: $D_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ $D_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

$|\nabla I| = \sqrt{(D_x I)^2 + (D_y I)^2}$

$\Phi(\nabla I) = \arctan(D_y I / D_x I)$



parametrize possible lines

edge px, detect peak

use counters to find peak point → best fitting line

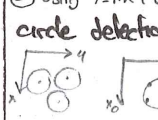
point ⟷ line, line ⟷ point

θ slope → use polar coord

Using $y = mx + c$ could be bad → line parallel to y → m had to parametrize

$\rho = x\cos\theta + y\sin\theta$

sample pointcurves

Circle detection


center

$(x_0, y_0, r) = (x - x_0)^2 + (y - y_0)^2 = r^2$

shift/rot invariant, not scaling

Detect corner points (Harris)
maximal difference with neighbors

$M = \begin{bmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{bmatrix}$

local displacement sensitivity $S(\Delta x, \Delta y) = \sum_{(x,y) \in \text{window}} (f(x,y) - f(x+\Delta x, y+\Delta y))^2$

↳ use linear approx $\approx (\Delta x\ \Delta y)\, M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$

desireable prop: = accurate localiz, robust/noise, invariance/shift/rot/scale/bright)

feature ∉ extraction SSD $\approx \Delta^T M \Delta$ (min for $\|\Delta\| = 1$)

use eigenvalues on structure matrix

- ellipse (EV) rotate but shape (EW) remain the same
- $\lambda_1 \gg \lambda_2$ edge
- λ_1, λ_2 both large corner

Cornerness $C(x,y) = \det(M) - k \cdot (\text{trace } M)^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)$

NOT scale inv. coz $(\Delta x \Delta y)$ is const. → detect corners that fit in w. otherwise corners detected as edges

4a. FOURIER TRANSFORM

Aliasing → can't shrink img taking 2nd px → errors (sampling) — small, phonem → fast — bigger, slower
differentiation expresses noise

fourier transf: new basis
$$F(g(x,y))(u,v) = \iint_{R^2} g(x,y)\exp(-i2\pi(ux+vy))\,dx\,dy$$
basis

$F = Uf$ vectorized img; fourier transf. base

linear transf. $f() \to \vec{v}_\phi$

transformed img

basis of F trans are eigenfk of lin.syst $e^{-i2\pi(ux+vy)}$ $F^{u,v}(x,y)$

Basis

Phase of complex transform important, dominant

Magnitude (very similar in natural img) we modify this

inverse (synthesis) $f = U^{-1}F$ high freq = small details | scale ↓ ↔ transform ↑

sin, impulse/delta, comb, box (gauss)

spatial domain — frequency

Convolution theorem * convolution · product
$F \cdot G = U(f * g)$
$F * G = U(f \cdot g)$ f,g funct | F,G FT
sampling | filtering

Sampling continuous → discrete $f() \to \vec{v}$
$$Sample_{2D}(f(x,y)) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f(x,y)\,\delta(x-i,y-j) = f \Sigma\Sigma \delta$$

FT of sample $F(Sample) = \ldots = \ldots ** \ldots = \sum_{i=-\infty}^{\infty}\sum F(u-i,v-j)$ **delta fk**

→ see CG 4.2 diagram

signal —FT→ magnitude; mult ↓ copy&shift (convolve); inverse FT ← cut out (box mult)

low sampling freq = aliasing: apply low-pass filter first (suppress high freq, the problem)

multiply FT with gaussian (not box, sinc because it has ∞ support)

Nyquist sampling thm sampling freq $\omega_s \geq 2\omega$ highest freq | if not, must bandlimit before sampling (low pass filter)
$\omega_N = 2\omega_{max}$ (rate)
n freq $= \frac{1}{2}\omega_s$ → produces no aliasing (if $\omega_{max} < n_{freq}$)

2DFT compute as seq of 1D FT
$$F^{u,v}g(x,y) = \iint \ldots dx\,dy = \int\left(\int g(x,y)e^{-i2\pi(ux)}dx\right)e^{-i2\pi(vy)}dy$$

FFT (fast FT)

reconstruction filters · square · bilin interp · gauss · perfect

Motion blur each light dot → short line along x1 axis
$$h(x_1,x_2) = \frac{1}{2\ell}(\theta(x_1+\ell)-\theta(x_1-\ell))\,\delta(x_2)$$

Restoration inverse kernel $\mathcal{H}$, $(\tilde{h} * h)(x) = \delta(x)$
$F[\tilde{h}] \cdot F[h] = 1$
$f(x) \to \boxed{h(x)} \to g(x) \to \boxed{\tilde{h}} \to f$
for motion blur: $F[h](u,v) = \ldots = \frac{\sin(2\pi u l)}{2\pi u l}$

⊖ vanishing frequencies cannot be recovered (V/2ℓ)

Space-hue super resolution
2D discrete signal (es digital img) → use Σ & 0 padding
lens + fix = low pass filter
$D * H * G$
$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi\left(\frac{x}{M}u + \frac{y}{N}v\right)}$$

5a. UNITARY TRANSFORMS

Linear Operator $O[\cdot]$
$O[a_1\cdot f_1 + a_2\cdot f_2] = a_1 O[f_1] + a_2 O[f_2]$

img → mat $N \times L = \hat{f}$ → vector

linear img processing alg $\vec{g} = H\vec{f}$

$\|\vec{f}\| = \sqrt{\sum_{MN\times MN}}$

unitary transform $A^{-1} = A^*(= A^H)$ real values → orthogonal

H: separate salient features, sparse g as result

$\vec{c} = A\vec{f}$

$\|c\| = c^H c = f^H A^H A f = \|f\|^2$ → coordinate rot (lengths conserved) (angles conserved)

img collection $F = [f_1 f_2 \ldots f_n]$

$R_{ff} = E[f_i \cdot f_i^H] = \frac{F\cdot F^H}{n}$ (img collection auto correlation f(k))

$R_{cc} = E[c_i \cdot c^H] = A R_{ff} A^H$ //auto cov matrix

cov matrix $C_X = \frac{1}{n}XX^T$

mean sq. values (average energies) on diagonal $E[c_i^2] = (R_{cc})_{i,i}$

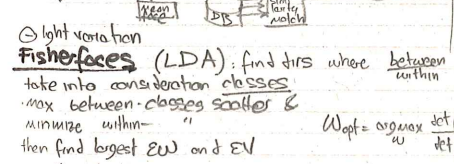
eigenmatrix of autocorrelation $M_{n\times n}$
Φ of R_{ff} : unitary, cols: eigenvectors of R_{ff}
$R_{ff}\Phi = \Phi\Lambda$ $\begin{bmatrix}\lambda_0 & & \\ & \ddots & \\ & & \lambda_{n-1}\end{bmatrix}$ diag.matrix of eigenv $\lambda_i \geq 0$

Karhunen-Loeve Transform (PCA)
$A = \Phi^H$, Φ cols sorted with decreasing λ's
$R_{cc} = \Phi^H R_{ff} \Phi = \Phi^H\Phi\Lambda = \Lambda$
most energy packed in first j coefficients $\forall j$ → mean sq. error minimized with first j comp.
Lagrangian cost fk $R_{ff}a_j^* = \lambda_j a_j^*$ correlated → uncorrelated, most in first comp. → reduce redundancy, discard repeated info
$p_{X=Y}$ $\forall m \forall d\, m_X \neq d_{m_Y} \forall c \in \mathbb{R}^k$

Eigenimages $\boxed{f = A^H c}$ superposition of basis img | Eigenimages = basis img = eigenv of R_{ff}

Recognition (faces)
expensive → everything against everything → reduce space dimension $\mathbb{R}^N \to \mathbb{R}^j$
look at euclidian distance → best match wins $\operatorname{argmin} D_i = \|I_i - I\| \cong \operatorname{argmin}\|p_i - p\|$
project to suitable space: S/N is improved
approximate $\hat{I} \cong \sum_{i=1}^{k} \varepsilon_i p_i$
$c = V^H(f-\mu)$ compress | decompress $f = Vc + \mu$
$\hat{I}_i = I_i - \bar{I}$, $\bar{I} = \frac{1}{n}\sum I_i$
$\Omega_1 = V^H(I_1 - \bar{I})$
$\Omega_2 = V^H(I_2 - \bar{I})$

Eigenface average face + lin comb of eigenfaces (8d space: recognise + generate) $\Sigma = \frac{1}{n-1}\sum \hat{I}_i \hat{I}_i^T$

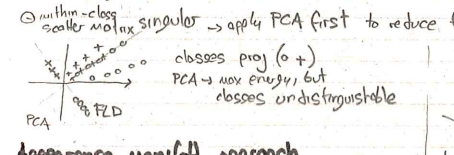


use SVD
Σ U V^T ≅ U_k Σ_k V_k^T

⊖ light variation

Fisherfaces (LDA): find dirs where between/within ratio of variance is max
take into consideration classes
max between-classes scatter & minimize within-
then find largest ΣW and ΣV
$W_{opt} = \operatorname{argmax}_W \frac{\det(WR_BW^H)}{\det(WR_WW^H)}$ $R_B w_i = \lambda_i R_W w_i$
$I_{sec} = \mu_k + c_k$
$c = E_k^T(I_{sec} - \mu)$

⊖ within-class scatter matrix singular → apply PCA first to reduce to L-c dimensions



Illumination $f(x,y) = a(x,y)(\vec{e}^T\vec{n}(x,y))L$ albedo, light intensity
lambertian surface: he in 3D space (different illum)
eliminated with fisherfaces

Appearance manifold approach
Vary sample viewing conditions

PCA used for: · lossy compression: that keeps much info by proj. · face recognition → sampling & viewing on different · img. info most important in PCs. · face detection

5b. PYRAMIDS & WAVELETS ③

Scale-space representation
signal $f(x)$ → parametric family $f^t(x)$, fine scale info suppressed

Pyramid N×N, N/2×N/2, 4×4, 2×2, 1×1
lvl 0 — approx filter → down sample → lvl 1 approx — up sample 2f, interp → reduction residual

· correspondence · edge tracking · detail / comp cost — fine ↔ coarse
es. CMU face detection

Gaussian P
gauss * gauss = gauss / smooth & sample
low pass → redundant representation

Laplacian P
save difference between levels
band pass f → each lvl has different spatial freq
last layer is the same — largely → always reconstruction

LoG lapl of gauss, laplacian
DoG difference of gauss

Compression quantization
g_0 —gauss→ Σ —L_1→ c_1 → reconstr. gauss
g_0 —gauss→ Σ —L_0→ c_0 → reconstr. lapl
original img

Oriented pyramids (OP)
LP orientation indep
→ apply oriented filter to determine orientation at lvl
→ represent img info at particular scale & orientation
layer 1, 2nd comp. layer 1, rot. diff., coarse, fine
LP, OP levels, LP, basis, extra

Wavelet transform finite wave
recursive application of 2-band filter bank to previous lowpass-band = octave band splitting

Haar transform H
· sample · real & orthog · 2D basis img of transform 8×8

Sub-op: scaling → info @ ≠ freq; transl → @ loc

filtering & down-sampl

Successive W/ subband decomp Lifting
on ≠ lvl → capture freq transitions save → loc

sampling
$s(t) = \sum_{n=-\infty}^{\infty}\delta(t-nT)$ $S(f) = \frac{1}{T}\sum_{n=-\infty}^{\infty}\delta(f-\frac{n}{T})$

$\sin\alpha = \frac{1}{2i}(e^{i\alpha} - e^{-i\alpha})$ $\int_{-\infty}^{\infty} e^{-i2\pi ux}dx = \delta(u)$
$\cos\alpha = \frac{e^{i\alpha}+e^{-i\alpha}}{2}$ $(ux+vy)\,dx\,dy = \delta(u,v)$

Fourier Trans. X(f) of signal x(t)
$F(x(t)) = X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi i f t}dt$ $(\omega = \frac{1}{T})$

STEPS PCA:
1) standardize data
2) obtain ΣV and ΣW from cov/corr matrix, or perform SVD
3) sort ΣW in desc order, select k biggest ($k \leq d$ new subspace)
4) construct proj mat W from selected ΣV
5) transform original dataset X via W to obtain k-dim subspace Y
POT: build LP of img at lex, then apply oriented filt at each lvl.
This yields img info at particular scale & orientation.

$\cos(\omega t) \to \pi[\delta(\omega-\omega_0) + \delta(\omega+\omega_0)]$
$\operatorname{sinc}(x) = \frac{\sin x}{x}$
$h(t) = (f*g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$
$\sin(2\pi\omega_0 t) \Rightarrow \frac{1}{2i}(\delta(\omega-\omega_0) - \delta(\omega+\omega_0))$
$(f-\delta)*\alpha = f*\alpha - \delta*\alpha = -(\alpha - f*\alpha)$
$\mathcal{F}[f, S_{2\Pi}(t)] = F^{(\omega)} * S_{2\Pi}(\omega) = \int_{-\infty}^{\infty} F(\tau)\cdot S(\omega-\tau)d\tau = \int \ldots S = \frac{1}{2\pi n=-\infty}\sum F$

PCA optimal ⊜ cone proof
$b = \frac{1}{j}c$ / $\frac{1}{j}$ only first j comp
$\varepsilon = \operatorname{tr}(R_{bb}) = \operatorname{tr}(\frac{1}{j}R_{cc}) = \operatorname{tr}(\frac{1}{j}A R_{ff} A^H_j) =$
$= \frac{1}{j}\sum_{k=0}^{j-1} a_k^T R_{ff} a_k^* \to$ lagr cost fk
$L = \varepsilon + \sum_{k=0}^{j-1}\lambda_k(1 - a_k^T a_k^*) = \Sigma a_k^T R_{ff} a_k^* + \Sigma\lambda_k(1 - a_k^T a_k^*)$
differentiate L wrt a_j yields
$R_{ff}a_j^* = \lambda_j a_j^* \; \forall_{j<J}$

6a. OPTICAL FLOW

Gestalt's law perception
· grouped · proximity · similarity · common fate

Where to pixels move to

1. Brightness constancy
OF = apparent motion of brightness patterns | proj of 3D velocity on img

$x(t)$, v; $y(t+dt)$; z (depth) plays role (closer appears faster)

no motion but change light → $OF \neq 0$

uniform, rotating sphere $O.F. = 0$

estimate OF needs
· brightness const · small motion · spatial coherence (move like N)

motion field → **NF**

$a_1 \to a$ **IBC**

$I(x,y,t) = I(x+u, y+v, t+1)$
$I_x u + I_y v + I_t \cong 0$
$I_x = \frac{\partial I}{\partial x}$, $v = \frac{dy}{dt} \cong x'-x$ (new-old)

Aperture probl: 1 eq 2 unknowns (dof 1 constraint) → smoothness: linearize with taylor
$\nabla I \cdot \vec{v} + I_t = 0$

gradient constr defines a line in (u,v) space

normal flow **NF**

aperture shape determines perception of mvmt

$v_n = -\frac{I_t}{|\nabla I|}$

OF computed for pt along a line/feature → can't determine exact loc, only NF

Regularization (Horn & Schunck) (smoothness) add normal smooth. constr
$e_s = \iint (u_{x+1}-u_x^2) + (v_{x+1}-v_x^2)\,dx\,dy$ min $e_s + \lambda e_c$
$e_c = \iint (I_x u + I_y v + I_t)^2\,dx\,dy$
$F = \lambda^{-1} + \ldots$

Lucas-Kanade: integrate over patch
more eq per px: spatial coherence constr: single $\vec{v}$ for all px in img patch $\begin{pmatrix}u\\v\end{pmatrix}$ assumption
$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{n\times n}) & I_y(p_{n\times n})\end{bmatrix}\begin{pmatrix}u\\v\end{pmatrix} = \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{n\times n})\end{bmatrix}$ $0 = I_t(p_i) + \nabla I(p_i)\cdot\begin{pmatrix}u\\v\end{pmatrix}$ ⇒ least squares
solution: $(A^TA)d = A^Tb$
A^TA $\begin{bmatrix}\Sigma I_x I_x & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y I_y\end{bmatrix}$ A^Tb $-\begin{bmatrix}\Sigma I_x I_t \\ \Sigma I_y I_t\end{bmatrix}$

solvable (A^TA) invertible
λ_1, λ_2 not too small (well-conditioned) — λ_1/λ_2 not too large (ill-conditioned)

large motion (> 1 px) → iterative refinement · coarse/fine est
no spat coherence → segmentation

es: visual inertial odometry (quadcopter)

Iterative refinement
estimate $\vec{v}$ per px with one iteration of L-K
warp one image towards the other using the estimated flow field
iterate (to refine)

Coarse to fine
not always brightness linearization works ⇒ reduce res using pyramids
warp → refine

linearization of brightness is suitable only for small displacement

Parametric motion mls (Global)
⊕ more constrained sol
2D · transl · affine · quadr
3D ...
$E(h) = \sum_{x\in R}(I(x+h) - I_0(x))^2$ $h = \begin{pmatrix}\delta x\\ \delta y\end{pmatrix}$ (A') (A'x) (A'y)

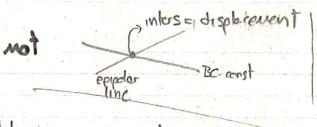
affine: move, scale, rot, warp
$A = \begin{bmatrix}a & b\\ c & d\end{bmatrix}$
$A' = \begin{bmatrix}a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9\end{bmatrix} = \boxed{R\ t}$

affine motion $u(x) = a_1 + a_2 x + a_3 y$ → $v(x,y) = a_4 + a_5 x + a_6 y$
1 px → 1 constr
6 unknowns ∑ min 6 px

KLT → good features to keep track of

3D model Ω rot, T transl, Z depth...

Residual planar parallax not
↳ results in epipolar field



Epipolar geometry

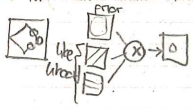
stereo vision

SSD tracking
large displacements → template match
small area as template
look in search area, use match measure (corr/error)
- discrete search → best estimate
- gradient method → lin. intensity and solve

⊕ video compr

$$I_0(x,y) = I(x,y)$$
$$I_1(x+v_0, y+v_0) = I(x,y) + \eta_1(x,y)$$

Bayesian flow
low lvl human motion illusion → explained with uncertainty model to lucas-k

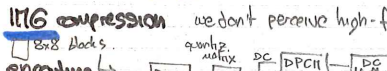


$$E(u,v) = \sum_{x,y} (I(x,y) - I_1(x+u+v))^2$$
$$= \sum_{x,y} (I(x,y) - I(x-u_0+u, y-v_0+v) - \eta_0)^2$$
$$(x,y)$$

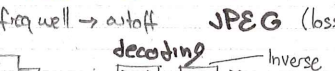
6b. VIDEO COMPRESSION

JPEG compression
we don't perceive high-freq well → cutoff — JPEG (lossy)

encoding



decoding



DCT (discrete cosine transform) – block based
- variant of DFT (Fourier)
⊕ real numb / fast implem

Block size: small	large
• faster	• better compr.
• better correlation	of large regions

B(0,0) 1st coeff is DC component = avg intensity

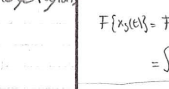
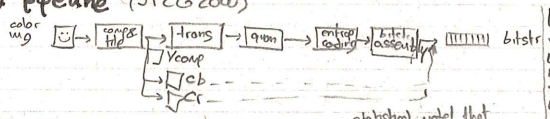


image compr.: concentrate most img info in low freq
- lose high freq (unimportant), cut B(u,v) @ bottom right
- toward high freq

Entropy coding – Huffman
encode frequent symbols w/ less bits
average length $H = -\sum p(s) \log_2 p(s)$

	prob	code	binary fraction
z	0.5	1	= code word as fraction 0.____ (ptr to interval)
x	0.25	01	
y	0.12..	001	
w	0.12..	...	

encoding pipeline (JPEG 2000)

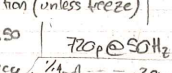


VIDEO compr
perception of motion
human visual syst → sensitive motion (<24Hz) — flicker up to >60Hz
no notice distortion (unless freeze)

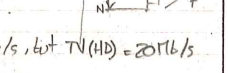
interlaced
2 temporally shifted img ⇒ increase freq 25→50, reduce spatial res

720p@50Hz > 1 Gb/s, but TV(HD) = 20Mb/s

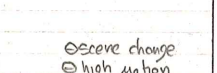
Lossy: take advantage of redundancy
- spatial correl (in px)
- temporal (frames)

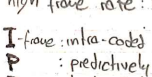


high frame rate: **temporal** redundancy
- transform/subband method
- predictive methods (based on previous/next frames, only motion & changes)

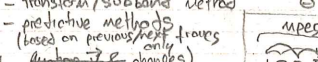


I-frame: intra-coded (indep)
P : predictively (dep of previous)
B : bi-dir predicted (" prev & future)





video compressor



MC (motion compensated) prediction
better than frame difference

Block matching mot. estimation ME
(es 16×16) → describe motion
alg
① blocks
② find best match
candidates: es all within 32×32 px (search area)

ref. f — curr. f

$f(n_1, n_2, k_{curr}) = f(n_1 - mv_1, n_2 - mv_2, k_{ref})$
motion vec (mv_1, mv_2)

metric: $MSE = \sum_{block} (f - \hat{f})^2$ (mean sq. e)
$MAE = \sum |f - \hat{f}|$
mvf = mv field

Search alg:
- full search : exam all
- partial (fast) " : exam subset

1: ±4
2: ±2
3: ±1
±7 search area, $2^4/4^2$ tries
best match @ each step

sub-px motion
mv not integer
- must spatially interp → fractional mv, es half-px ME (mpeg 1/2/4)
↳ bilinear interp
↳ 4: ± 1/2 (or not)

⊕ avg effect → reduce error & compression

block matching
⊖ size? search range?
⊕ std/robust/simple
⊖ assume translation only
blocking artifacts

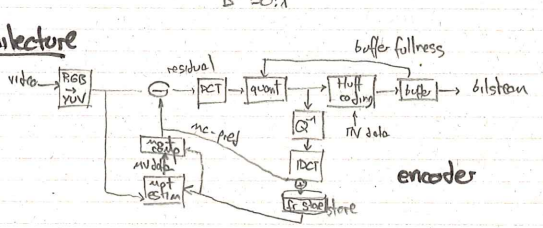
bidirectional MC prediction
use prev/future/avg/none to predict
↳ make I

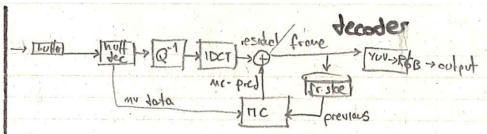
- MC pred yields
  - mv
  - mc pred error/residual (code normally)
- if fail, code I frame

GOP (group of pic)
start I, end before I: B (open) → still need to look forward
P (close)

	compression
I	7:1
P	20:1
B	50:1

Video compr. Architecture
exploit redundancy
- temporal (MC)
- spatial (DCT)
- color (col space conv)





Standards
goal: interoperability (comm devices) costs/technology (techn, not implem)

enc → bits → dec
↳ only this is defined, and decod process

- JPEG / 2000
- H. 261/263/264
- MPEG 1/2 (G) / (AVC)
- ~~frame~~ obj based coding
- MP3

PSNR: (peak S:N ratio)
$e(v,h) = \hat{x}(v,h) - x(v,h)$ per 1 px

$e_{use} = \sqrt{\frac{1}{NM} \sum_{vi} \sum_{hi} e^2(v,h)}$
↳ mean-square error

↳ $= \frac{\max\{x^2\}}{e^2_{use}} \equiv 10 \log_{10} \left(\frac{(2^K)^2}{e^2_{use}} \right) dB$

Exercise ② Nyquist Theorem states that sampling frequency must be at least twice the highest frequency of the signal $\omega_s > 2\omega_h$. Use Fourier tansform to show why Nyquist sampling theorem should be (For simplicity you can use 1D function instead 2D). ($\omega_s = 1/\tau$)

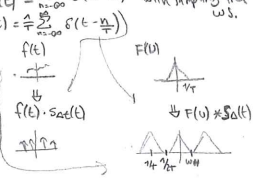
- Define continuous function as f(t) and the sampling function $s_{\Delta t}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\tau)$ with sampling freq ω_s
  $s_{\Delta t}(t) = \frac{1}{\tau} \sum_{n=-\infty}^{\infty} \delta(t - \frac{n}{\tau})$

The Fourier transform of the sampled function can be derived as:

$$\mathcal{F}(\omega) = \mathcal{F}\{f(t) s_{\Delta t}(t)\} = F(\omega) * s_{\Delta t}(t)$$
$$= \int_{-\infty}^{\infty} F(\tau) s_{\Delta t}(\omega - \tau) d\tau$$ ← Fourier transf. of original signal f(t)
$$= \int_{-\infty}^{\infty} F(\tau) \frac{1}{\Delta\tau} \sum_{n=-\infty}^{\infty} \delta(\omega - \tau - \frac{n}{\Delta t}) d\tau$$
$$= \frac{1}{\Delta\tau} \sum_{n=-\infty}^{\infty} F(\omega - n\omega_s)$$

- $F(\omega)$ cannot overlap its neighbours, i.e. $F(\omega - \omega_s)$ and $F(\omega + \omega_s)$.

Thus ω_s should be larger than $2\omega_h$ where ω_h is highest frequency. → if this fails → aliasing



solution: filter signal with a low-pass filter to reduce its high freq. to acceptable levels.

▸ Let $x_s(t) = x(t) s(t)$ → use conv. theorem → derive Fourier transf. in terms of x(t):

$$\mathcal{F}\{x_s(t)\} = \mathcal{F}\{x(t) \cdot s(t)\} = \mathcal{F}\{x(t)\} * \mathcal{F}\{s(t)\} \overset{\text{def. conv}}{=} \int_{-\infty}^{\infty} \mathcal{F}\{x(t)\} \cdot S(f - \gamma) d\gamma$$
$$= \int_{-\infty}^{\infty} x(t) \cdot \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta(f - \gamma - \frac{n}{T}) d\gamma = \frac{1}{T} \sum_{n=-\infty}^{\infty} F(f - \frac{n}{T})$$

▸ DIRAC-DELTA $\int_{-\infty}^{\infty} \delta(x) dx = 1$, $\int_{-\infty}^{\infty} f(x) \delta(x) dx = f(0)$, $\delta(x-a) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ip(x-a)} dp$
$\delta(x - x_0) = \frac{1}{|k|}$

▸ implicit $x^2 + y^2 = 0$ ⊕ inside/outside test ⊖ compex find all ptr in shape ⊖ difficult for complex shapes
▸ explicit $(r\cos\theta, r\sin\theta)$

▸ left $p' = Mp$. If n is normal then: $n' = (M^{-1})^T n$ (of trans. plane, of pri. plane)
▸ Point in triangle test: $P_i(x_i, y_i)$ $dx_i = x_{i+1} - x_i$, $dy_i = y_{i+1} - y_i$, $G(x,y) = (x - x_i) dy_i - (y - y_i) dx_i = A_i x + B_i y + C_i$
= 0 pt on edge, > 0 outside edge, < 0 inside

▸ Level Set Method
- implicit have some nice feature (es near/split), but hard to desc. complex shape in closed form
- alternative store a grid of value approximating func.
- surface is found where interpolated values equal zero
- provides much more explicit control over shape

▸ Filtering texture
- Minification: area of screen px maps to large region of text (filtering req.: averaging)
- one texel corresponds to far less px or even to many (⇒ aliasing)
- magnification: area of screen px maps to tiny region of text (interpolation req.)

7.b TEXTURE

We don't use histograms for efficiency reason → filter on wavelet or Gabor filter

One way of representing using Laplacian Pramid:
- build Lapl. pyramid of text and apply a # of oriented filters to each level. This represents img. info at a particular scale and orientation

Representing tex: key issue
⚠ pre-attentive tex discrimination ⊥ L
↳ stylized subelements with meaningful repetition
↳ find & represent statistics → $\bar{x}, \sigma, \ldots$
↳ use filters & magnitude of response (normalized correlation)
↳ spots & oriented bars

Analysis/segmentation (DISCRIMINATIVE)
compare (& save)

Synthesis (GENERATIVE)
input img → ∞ texture (large, constraints, ..)

representation:
- form oriented pyramid
- square output
- take statistics of responses

Gabor filter
different scales/spatial freq

Synthesis
use img as source of probability
choose P_x val by matching n & fill in

Histogram

→

lose spatial info
→ intensity probability distrib

co-occurrence matrix
= prob distr of intensity pairs

Tex transfer
take & paint texture onto obj
HARD (tex - shape)
- boundary & shading

extra constr.: similar to underlying img

Zolesny-Van Gool (view-dep)
n-analysis iterations

Efros-Leung $P(p|N(p))$
search src img for similar neighborhood
pick 1 match at random
(spatial constraints)

explicit prob tables infeasible

Chaos mosaic
- tile input img
- pick random block & place random
- smooth edges

⊖ structured tex don't work
⊕ foliage

Img Quilting (window)
unit of synthesis = block — $P(B|N(B))$
faster: block by block (not P_x by P_x)

$(B_1 - B_2)^2 =$ DP alg to cut — **minimal err boundary cut**

alg:
- pick block size/overlap size
- synthesize blocks
- look for block to satisfy constraints (top & left)
- paste
- compute min err cut ⊖ fruits

1.2 GRAPHICS PIPELINE

in: Geometry, materials, Lights, Camera → GP → out: display colors

BRDF (4D fct) — bidir. refl. distr. fct

use simpler models (materials)

modeling trans (obj → world) → viewing trans (world → cam) → 2D clip (frustum; reduce; stability) → light, shading, texturing (compute color (light, mat, normals)) → proj to screen sp (3D → 2D) → scan conversion (interpolate vx → px (discretization of primitives))

vx | px

Nyquist thm: sample twice max frequency has infinite frequency → aliasing

Programmer's view

CPU → vertex Processing (vx op; transforms, lighting, flow control) → rasterization (NDC coords pts.; px op; screen space px) → Fragment Processing (shading, texturing, blending) → Display

vx shader → interpol → frag shader (programmable)

attributes /vx → vx shader → varying /vx → interpol → varying /frag → frag shader → color /frag

attributes → VS VS VS → varying interpolable varying → FS → color

Fragment ≠ pixel (frag | px) contains: position, depth, texture coord..

Code: vx shader, frag shader

G API

agess graphics HW independly, abstract complexity

low lvl: open GL, direct X (drawing commands)

high lvl: Renderman, ... (DAG for scene)

open GL: standardized graphics; similar to C; updated for HW; language bindings

double buffer (one render, one build)

libraries: glu.lib (main), glu.lib (utilities), glut.lib (toolkit), glext.h (extension)

windowed app → draw (GLU → open GL → windowing syst) | events (GLUT → events)

window management, user input

Code example application (display fct, vs, fs)

glDrawElements (x, count, vint, 0);

GL_POINTS, LINES, POLYGON, TRIANGLES, QUADS, TRIANGLE-STRIP, -FAN

- Luminous Flux F [lumen]: $F = c \cdot \int_{380nm}^{770nm} P(\lambda) V(\lambda) d\lambda$; Illumination E [lux]: $E = dF/dA$
- Luminous Intensity I [candela]: $I = \frac{dF}{d\omega}$; $F = \int I \, d\omega$ | Luminance Y [candela/m²]: $Y = \frac{d^2F}{dA \cos\theta \, d\omega}$; $I = \int Y dA$, $F = \iint Y dA d\omega$

2.1 LIGHT AND COLORS

what is light = EM radiation

spectrum (amplitude determines intensity)

frequency ν: 10^{16} 10^{14} 10^{12} 10^{10} 10^{8} 10^{6} 10^{4} 10^{2} 10 (Hz)

cosmic rays | γ rays | x rays | UV | IR | microwave | FM | AM radio waves | Long radio waves

10^{-16} 10^{-14} 10^{-12} 10^{-10} 10^{-8} 10^{-6} 10^{-4} 10^{-2} 10^{0} 10^{2} 10^{4} 10^{6} 10^{8}

high E

380 — 770 nm

E → Increasing wavelength (λ) (m)

light = mixture of wavelenght; SPD → spectral power distribution

each rays $P(\lambda) \to \infty$ dim vector (human can't process)

↳ project into subspace (discretize using cones) (different SPD might look same to us) metamer

some info lost

$x = x_1 n_1 + x_2 n_2 + x_3 n_3$

$x = (x_1 n_1) n_1 + \dots$

$xP = (\quad) + (\quad) + (\quad)$ — (leave term out for projection)

coord are continuous inner prod

$x_i = \int x(\lambda) n_i(\lambda) d\lambda$

$x^{3D} = (x_1 x_2 x_3) \to x^{\infty} = x(\lambda)$

$x^P(\lambda) = x_1 n_1(\lambda) + x_2 n_2(\lambda) + x_3 n_3(\lambda)$ (combination of primary colors)

Definitions

- Luminous Flux: relative spectral power density weighted by the relative spectral sensitivity integrated over the complete human visible spectrum.
- Luminous intensity: luminous flux emitted through a specific angle in space.
- Illumination: Luminous flux of a light source falling on a surface element.
- Luminance: Luminous intensity of a light source, considering direction & surface normal.
- Color Stimulus Specification: color coordinates in a color space approximating the relative spectral power density.

CIE Primary system (1931) comission internationale de l'Eclairage

measure Human color sensitivity: B 435.8 (B) 546.1 (G) 700 (R) nm

light src → filter, diffuser → pure spectral color (monochrom. light)

stimulus 2 | stimulus 1 (ref) (dimmer controls intensity)

$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} \bar{r}(\lambda_1) \dots \bar{r}(\lambda_n) \\ \bar{g}(\lambda_1) \dots \bar{g}(\lambda_n) \\ \bar{b}(\lambda_1) \dots \bar{b}(\lambda_n) \end{bmatrix} \begin{bmatrix} P(\lambda_1) \\ P(\lambda_2) \\ P(\lambda_n) \end{bmatrix}$ control dimmer to have same color

color matching functions; SPD of test light

use monochromatic test light $\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

Color matching as matrix mult

intensity for primaries

wavelength of monochromatic light that was on stimulus 1

negative matching value → add ref light to test light

→ some colors cannot be written as lin-comb (outside of gamut)

basis is arbitrary → perceptual system dependant

new basis → linear transf → same space

RGB (additive) → non neg. lin. comb of 3 primaries (spongiest)

color mixing not intuitive

not negative anymore

Cie XYZ

$x = \int_0^\infty P(\lambda) \bar{x} \, d\lambda$ (same Y, Z)

(X, Y, Z) quantifies any P(λ) stimulus

$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2.36 & -0.515 & 0.005 \\ -0.89 & 1.426 & 0.014 \\ -0.46 & 0.088 & 1.009 \end{pmatrix} \cdot \begin{pmatrix} \bar{r}(\lambda) \\ \bar{g}(\lambda) \\ \bar{b}(\lambda) \end{pmatrix}$

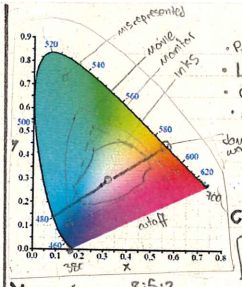
plot xy = all colors same brightness

Chromacity (x, y), Y = brightness (if mixing $Y_{12} = Y_1 + Y_2$)

$x = \frac{X}{X+Y+Z}$ ① $y = \frac{Y}{X+Y+Z}$ ② $z = \frac{Z}{X+Y+Z}$

$x + y + z = 1$

① from ②③: $X = \frac{Y}{y} x$ ③ from ②③: $Z = \frac{Y}{y}(1 - x - y)$



- Primaries along curved boundary (spectral colors) single wavelength
- Linear combin: line betw 2 pts
- Gamut: lin-comb of 3 pts
- : white pt @ $\begin{pmatrix} 0.32 \\ 0.34 \end{pmatrix}$ (theor (0.3)0.33) ↳ DCS

purple-line: non spectral purples (colors) (contain multiple wavelenghts)

other colors spaces

CMY (subtractive color systems, e.g. PRINTERS) from pos lin comb of the 3 primaries | K → adds black color

$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$ RGB → W; CMY → +K

YIQ (Y = luminance, I = in phase (orange-blue), Q = quadrature (purple-green))

⊕ natural colors & skin NTSC-US TV standard

⊕ compress save channels with more bandwidth

$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$

HSV value, lightness, brightness ⊕ user oriented - color picking; dimensions no longer primary

Hue → base color; Sat → purity

RGB → proj to hexagon (not lin. transf)

Code RGB → HSV

$MIN = \min\{R, G, B\}$, $MAX = \max\{\dots\}$, $V = MAX$

if $(MAX \neq 0)$ $S = \frac{MAX - MIN}{MAX}$ else $S = 0$

H: hue (V, S, R, G, B)!

Perceptually uniform color spaces

colors close together not necessarily similar visually → Mac Adams (ellipses)

Cie Lab, Cie Luv: ellipses are almost circular (non linear warps of CIE chart)

YUV (PAL): $Y(XYZ) = -R + -G + -B$; $C_b = B - Y$, $U = 0.49 C_b$; $C_r = R - Y$, $V = 0.88 C_r$

$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{bmatrix} 0.2126 & -0.7152 & 0.0722 \\ -0.0992 & -0.3504 & 0.4496 \\ 0.6929 & -0.6296 & -0.0633 \end{bmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix}$

white pt color: $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} x_r C_r & x_G C_G & x_B C_B \\ y_r C_r & y_G C_G & y_B C_B \\ (1 - x_r - y_r) C_r & (1 - x_G - y_G) C_G & (1 - x_B - y_B) C_B \end{bmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$

	R	G	B	find C_R ? $\begin{pmatrix} 0.950 \\ 0 \\ 1.089 \end{pmatrix}$
x	0.64	0.30	0.15	C_G
y	0.33	0.60	0.06	C_B

2.2 TRANSFORMATIONS

transf: affine, linear, projective, rigid; T, R, S, SH, PE, PA; perspective, parallel

P point, A matrix

linear map $A(\alpha x + \beta y) = \alpha Ax + \beta Ay$

Rigid $Ax + b$, $\det A = 1$; $A^{-1} = A^T$ → preserve vector length

Affine $Ax + b$

2D transformations

transl $P' = p + t$ $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$

scaling $P' = Sp$ $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

rotation of θ (z-axis): $P' = Rp$ $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

Homogenous coords

$p = \begin{pmatrix} x \\ y \end{pmatrix}$ $p' = Tp$ $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ affine maps are linear in homog. coord

$p = \begin{pmatrix} wx \\ wy \\ w \end{pmatrix}$ ∞ many repres (any w)

(line in 3D) affine plane

$S/R = \begin{bmatrix} & & 0 \\ & & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Shear $SH_x = \begin{bmatrix} 1 & a & \\ & 1 & \\ & & 1 \end{bmatrix}$ $SH_y = \begin{bmatrix} 1 & & \\ b & 1 & \\ & & 1 \end{bmatrix}$

combining

TRx = rotation followed by transl

	T	R	S
T	V		
R		V	
S		V	V

Commutativity: ~ 2D only (order matters)

$M_1 M_2 = M_2 M_1$

$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$ $\begin{bmatrix} I & + \\ 0 & 1 \end{bmatrix}$ $\begin{bmatrix} S & 0 \\ 0 & 1 \end{bmatrix}$

3D transformations

$p = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \to \begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix}$ hyperplane projection

$T = \begin{bmatrix} 1 & & & t_x \\ & 1 & & t_y \\ & & 1 & t_z \\ & & & 1 \end{bmatrix}$ $S = \begin{bmatrix} s_x & & & \\ & s_y & & \\ & & s_z & \\ & & & 1 \end{bmatrix}$

$R_x = \begin{bmatrix} 1 & & & \\ & \cos\theta & -\sin\theta & \\ & \sin\theta & \cos\theta & \\ & & & 1 \end{bmatrix}$ $R_y = \begin{bmatrix} \cos & & \sin & \\ & 1 & & \\ -\sin & & \cos & \\ & & & 1 \end{bmatrix}$ $R_z = \begin{bmatrix} \cos & -\sin & & \\ \sin & \cos & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$

right hand rule (all > 0 represent CW rotation of axes)

$R(\vec{v}, \theta) = \dots$ rot around axis $\vec{v}$

$SH_{xy} = \begin{bmatrix} 1 & 0 & sh_x & 0 \\ 1 & sh_y & 0 & \\ 1 & 0 & & \\ & & 1 & \end{bmatrix}$ $SH_{xz} = \begin{bmatrix} 1 & sh_x & & \\ & 1 & & \\ & sh_z & 1 & \\ & & & 1 \end{bmatrix}$ $SH_{yz} = \begin{bmatrix} 1 & & & \\ sh_y & 1 & & \\ sh_z & 1 & & \\ 0 & & & 1 \end{bmatrix}$

Coordinate syst

$p' = \begin{bmatrix} r_1 & r_2 & r_3 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} p = \begin{pmatrix} t + r_1 p_x + r_2 p_y + r_3 p_z \\ 1 \end{pmatrix}$

TR = change of basis (2 interpretations) same

$p = p_x x + p_y y + p_z z$

o object, w world, c camera

normals $P' = Mp \to n' = (M^{-1})^T n$ $\quad (A^{-1}) = (A^T)^{-1}$ $\quad (A^T x)^T = x^T A$

$p = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ satisfies $Ax + By + Cz + D = 0$ $\quad n^T p = 0$ $\quad n = \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix}$

$M = LT$, $(M^{-1})^T = L^{-1} T^{-1} = L T^{-1}$ (L: unitary)

Projection 3D → 2D: parallel, persp (vanishing pt)

make things appear smaller / bigger

parallel: in 3D not anymore in 2D (notice up-close) — general

persp: view along y; proj plane; PE

$x_p = (\theta/2) x$, $y_p = (\theta/2) y$

$M_{per} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & 0 \\ & & 1/d & 0 \end{bmatrix}$ $M_{ortho} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & 1 & \end{bmatrix}$ (ignore unwanted coord)

$M_{per}\, p = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} = \begin{pmatrix} xd/z \\ yd/z \\ d \end{pmatrix}$ 3D coord; depth in own space

Clipping planes

front pl, back pl; only obj in V drawn; deform V → fit in it

Quaternions

complex numbers in 3D 4D

$z = a + ib + jc + kd$ $\quad i^2 = j^2 = k^2 = -1$ $\quad ij = k$ $\quad ji = -k$ $\quad jk = i$ $\quad ki = -i$ $\quad ik = -j$ $\quad ijk = -1$

(can be represented as unit length) → infer a from unit vecto v

Properties (coefficients)

sum

$\|z\| = \sqrt{a^2 + b^2 + c^2 + d^2}$ • $z = s + \vec{v}$ (scalar + vector) • $\bar{z} = s - \vec{v}$ (conjugate) $\quad z\bar{z} = \|z\|^2$ $\quad z^{-1} = \frac{\bar{z}}{\|z\|^2}$

$\frac{z}{\|z\|}$ (unit) • $z_1 z_2 \neq z_2 z_1$ (not commute) $\quad \|q_1 q_2\| = \|q_1\| \|q_2\|$

real part of $q_1 q_2 = q_2 q_1$

$p = q p q^{-1}$ $\quad z z^{-1} = z^{-1} z = 1$

here p is in pure quaternion form (s = 0)

$R(\vec{v}, \theta) = \cos(\theta/2) + \sin(\theta/2) \frac{1}{\|v\|} (v_x i + v_y j + v_z k) =: q$ (q: unit quaternion)

⊕ easy + efficient

⊕ no gimbal lock

Open GL transformations

vertex (xyz1)^T → world view trans (1) → eye coord → proj (2) → clip coord → perspect division (3) → normalize device coord → viewport transform (4) → window screen coords

Model → World: $\begin{bmatrix} r_1 & r_2 & r_3 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} wx \\ wy \\ wz \\ 1 \end{pmatrix} = \begin{pmatrix} \text{left up dir eye} \\ 0 \quad 0 \quad 0 \quad 1 \end{pmatrix} \begin{pmatrix} cx \\ cy \\ cz \\ 1 \end{pmatrix}$

World → camera: left = 100^T, up = 010^T, dir = 00−1^T, eye = 000^T (use inverse matrix)

clipping: ① save computation ② memory consistency ③ no /0 (stability, close to cam)

② Parallel proj | glOrtho (left, right, bottom, top, near, far) — only pts inside are projected

Perspective proj | glFrustum (" ");

(same) | clipping: $p=\left(\frac{c_x}{c_w}, \frac{c_y}{c_w}, \frac{c_z}{c_w}\right)$ compare them with 1 if all smaller = inside frustum

③ normalized screen/device coords

depth test (dist along ray)

④ screen px

$f = \cot\left(\frac{\alpha}{2}\right)$

glViewport(o_x, o_y, w, h); glDepth Range (n, f)

3.1 LIGHTING & SHADING 1.

measuring light = counting photons **Radiometry** → study EM radiation, incl. light

$\theta = \frac{l}{r}$ [2π tot] $\Omega = \frac{A}{r^2}$ (4π tot) $A = 4\pi r^2$ $V = \frac{4}{3}\pi r^3$

$\omega = (\theta, \phi)$ zenith / azimuth

$w_x = \sin\theta\cos\phi$ $w_y = \sin\theta\sin\phi$ $w_z = \cos\theta$

$lat = \frac{90}{\pi}(\pi - \theta)$ $long = \frac{90}{\pi}\phi$

$dA = (r d\theta)(r\sin\theta\, d\phi)$ $d\vec{\omega} = \frac{dA}{r^2} = \sin\theta\, d\theta\, d\phi$ $\Omega = \int_{S^2} d\vec{\omega} = \int_0^{2\pi}\int_0^{\pi} \sin\theta\, d\theta\, d\phi = 4\pi$

Photons
x pos
$\vec{w}$ direction of motion
λ wavelenght

$E = \frac{hc}{\text{plot } \lambda}$ $h = 6{,}63 \cdot 10^{-34}\, m^2 \cdot \frac{kg}{s}$ Plank's const $[J = kg \cdot \frac{m^2}{s^2}]$

$c = 299{,}792{,}458 \frac{m}{s}$ (Light's speed)

Radiometry
flux $\Phi(A)$ [W] tot E passing though surface/space per time 1/s es: photons hitting wall per second/leaving lamp
Irradiance $E(x)$ [W/m^2] flux per unit area arriving at surface $\frac{d\Phi(A)}{dA(x)}$
radiosity $B(x)$: " es: photons hitting wall area / wall area
es photons reflecting leaving (dual of irradiance / antagonistic)

Lambert's Cosine Law
$E = \frac{\Phi}{A}\cos\theta$ (the more perpendicular, the brighter)

radiant intensity $I(\vec{w}) = \frac{d\Phi}{d\vec{\omega}}$ [$\frac{W}{sr}$] power (flux) per solid angle = directional intensity of flux
$\Phi = \int_{S^2} I(\omega)\, d\vec{\omega}$ es: power per unit solid angle emanating from lamp
$\Phi = 4\pi I$ | isotropic pt src (uniform in all directions)

Radiance $L(x, \vec{w}) = \frac{dI(\vec{w})}{dA(x)} = \frac{d^2\Phi(A)}{d\vec{\omega}\, dA(x)\cos\theta}$ [$\frac{W}{m^2 sr}$]
$dA^{\perp}(x, \vec{w})$ ray tracing, const along ray (others can be expressed with it)

Reflection models

BRDF (Bidirectional reflectance distrib. function) 4D (2in 2out) spectral characteristics
differential irradiance → a cone of directions around $\vec{w_i}$: $f_r(x, \vec{w}, \vec{w_r}) = \frac{dL_r(x, \vec{w_r})}{dE_i(x, \vec{w_i})}$
i = incident r = reflected (local illum. model) [1/sr]

reflection eq. $L_r(x, \vec{w}_r) = \int_{\Omega^2} f_r(x, w, w_r)\, L_i(x, \vec{w}_i)\cos\theta_i\, d\vec{w}_i$

complex reflection
anisotropy: property of being reflection dependent

SIMPLER REFLECTIONS (phong model) ⊕ more efficient, simpler ⊕ open GL

•Ambient light coming from all dir. independent of camera/light/surface orient
$I = I_a k_a$ (light source, material parameter (constant))

•Diffuse reflection directed light I_p dependent on • light pos • surface orientation independent on • camera (reflected in all directions)
$I = I_p k_d \cos\Theta = I_p k_d (N \cdot L)$
sum them → simple model!

Attenuation
due to spatial radiation $f_{att} = \frac{1}{d_L^2}$
model in open GL: $f_{att} = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right)$
$I = I_a k_a + f_{att} I_p k_d (N \cdot L)$

•Specular reflection
depends and viewing ray
$R = N\cos\Theta + S$
$= 2N\cos\Theta - L = 2N(N \cdot L) - L$
$\cos\alpha = R \cdot V = (*) \cdot V$

A+D+S approx specular reflection by cosine powers

$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{i=1}^{m} f_{att_i} I_{p_i} [k_d O_{d\lambda}(N \cdot L_i) + k_s (R_i \cdot V)^n]$ (lucentezza)

$O_{d\lambda} = \frac{1}{3}$ multiple light sources | k_a, k_d, k_s material dependant

Halfway vector (faster) $\cos^n\beta = (N \cdot H)^n$ $H = \frac{L+V}{\|L+V\|}$ (compute for all RGB components)

3.2 LIGHTING & SHADING 2.

Shading models
• **flat shading** (1 color/primitive) • **Gouraud shading** (lin. interp of intensities) screen space
• phong shading (lin. interp of vx normals) ← more sophisticated, per px eval (−) performance
Lobj space

Gouraud
• calc face norm
• calc vx norm (average)
• evaluate ill. mod. per vertex
• interpolate vx bilinear on scan line
Problems
⊖ perspective distortion
⊖ orientation dependence
⊖ shared vertices

$I_a = I_1 - (I_1 - I_2)\frac{y_1 - y_s}{y_1 - y_2}$
$I_b = (I_1)(\frac{\quad}{y_1 y_3})$
$I_p = I_b - (I_b - I_a)\frac{x_b - x_p}{x_b - x_a}$
quality dependant on primitive size

Phong shading / refl model
barycentric interp of normal
$n_x = \lambda n_a + \lambda n_b + \lambda_c n_c$
$\Sigma\lambda = 1$ $(\lambda_a + \lambda_b + \lambda_c)$
$\lambda_a \cdot a + \lambda b \cdot b + \lambda c \cdot c = x$
$\lambda_a = \frac{\text{area opposite triangle}}{A\,tot}$
properties: $x = a \to n_x = n_a$
Problems: ⊖ normal not defined / representative

transparency composite transparent obj
Alpha-blending RGBA (linearizes exponential intensity attenuation)
α = absorption, intensity filtered by P_1
$I_\lambda = I'_{\lambda_1} + I'_{\lambda_2} = I_{\lambda_1}\alpha_1 \Delta t_1 + I_{\lambda_2} e^{-\alpha_1 \Delta t}$
I_{λ_1} emission
$\alpha_2 = 1$ (opaque)
Δt (thickness)
linearize $I'_{\lambda_2} = I_{\lambda_2}(1 - \alpha_1 \Delta t)$
$\Rightarrow I_\lambda = I_{\lambda_1}\cdot\alpha_1 + I_{\lambda_2}(1 - \alpha_1)$

rendering order needed: sorted polygon traversal back to front rendering

depth peeling solution
• multiple passes
• each pass renders next closest frag → 2nd 1st

other lighting models
• cook-torrence (metal/specular)
• ashikh min (anisotropy)

4.1 GEOMETRY & TEXTURES 1. ⑥

0D 1D 2D 3D (womfold)

Applications
• game, movies (mesh)
• engineering (CAD); polynomes
• medicine/biology MRI → 3D volumetric data
• architecture (statics)

Sources
real world: scanning, kinect
• point clouds (noisy)
digital: 3D modelling (blend) / vectors
procedural modelling (grammar, rules)

considerations
• storage • acquisition • editing
• rendering • creation

Geometry representation
Polygonal meshes • boundary • geometry + connectivity
fast rendering, dedicated GPU pipeline

Polygonal mesh
vertices $v_0, v_1, \ldots v_{n-1}$ | edges $\{(v_0, v_1), \ldots (v_{n-2}, v_{n-1})\}$
planar, non-self-interesting
Mesh = set of connect poly gons $M = \langle V, E, F \rangle$ → faces

properties
• all edges belong to polygon
• intersection of polygons is: - empty - vertex - edge
deg (v) | Boundary: all edges that belong to polygon

$v - e + f = 2$
$e \leq 3v - 6$
$f \leq 2v - 4$

Manifold surface locally homeomorphic to a disk
if closed divides space in 2 (real world data often non-manifold)
not allowed

Mesh DS
(store geometry & topology) + attributes (normal, color, ...)
vx: location; edges/faces
oo: rendering geometry queries (es vertices of F)
modifications (es remove V)

triangle list

triangles		
vx ind	vx ind	vx ind
(x_0, y_0, z_0)	(x_1, y_1, z_1)	...
(,)		

⊕ simple
⊖ no connectivity
⊖ redundant
STL file

Indexed face set

triangles		
v ind	v ind	
0	1	2
1	2	...

vertices	
ind	coord
0	(x_0, y_0, z_0)
1	
2	

⊕ efficient
⊕ not-redundant
⊕ connectivity
⊖ expensive query + modification
OBJ/OFF/WRL files

Mesh textures enhancing realism / increasing geometric complexity without
mapping texture + geometry = rendering
ISSUES
• mapping
• anti-aliasing, filtering
• level of detail (LOD)
geom ↔ texture (1:1 map)
$x(u,v), y(u,v), z(u,v)$
parametrization $\mathbb{R}^2 \to \mathbb{R}^3$
es sphere $\begin{bmatrix} u \\ v \end{bmatrix} \to \begin{pmatrix} \sin u \cdot \sin v \\ \cos v \\ \cos u \cdot \sin v \end{pmatrix}$

desiderable properties low distortion
bijective mapping
efficient comput
finding cuts
texture atlas

4.2 GET 2.

texture filtering
Sampling | Reconstruction
multiply | original fct | sampled fct | repethon | spatial domain | frequency domain (magnitude)
sinc | (cutoff) | filter function | reconstructed funct

Aliasing
less sampling → aliasing
middle space
not the same we started with!
autoff = convolution in spatial dom

→ **low pass filter** to avoid aliasing
multiplication | filtered function

Aliasing
• jaggy edges
• 'more' patterns
• less of detail
high frequency

structured vs unstructural noise
we like it better (retina discretizes light, but very well → no Aliasing)

isotropic 2D gaussian $f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$
anisotropic $= \frac{1}{2\pi\sigma_x\sigma_y} e^{\left(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}\right)}$
(spatially-varying tex filter)

tex | screen space

light maps • simulate effect of local light src ⊕ precomputed / dyn. adopted tex + light map = render

Environment maps
• rendering reflective obj efficiently
• normal-color correspondance
tex coord computed intersecting reflected ray with sphere (use photos as texture)
→ cube allows simpler computation

Bump mapping • perturb surface normal according to tex (grayscale) height map
• small scale geometry ≠ positional perturbation
$n = \frac{\partial p}{\partial u} \times \frac{\partial p}{\partial v} = P_u \times P_v$ u, v parameter domain
$p' = p + \frac{bn}{|n|}$ $n' = \frac{\partial p'}{\partial u} \times \frac{\partial p'}{\partial v}$ $\frac{\partial p'}{\partial u} = \frac{\partial p}{\partial u} + \frac{\partial (bn)}{\partial u |n|}$
$n' = n + b_v(n \times P_u) + b_u(n \times P_v)$ (use FD method)
$b_u = \frac{b - a + c - d}{2\Delta u}$ $b_v = \frac{d - a + c - b}{2\Delta v}$
⊖ no silhouette bumps
⊖ no self occlusion / shadows
cube map (simulate height map)

Normal mapping (RGB to simulate direction as well)
→ shading / fake illumination
• tangent space (deformable, blue/purple)
• obj space (faster/rainbow)

Displacement mapping • pertub geometry based on texture

Procedural / solid textures
• perlin noise
• gabor noise (⊕ control over spectral prop)

Mid-mapping
• store down-sampled version of tex
• use low-res for far-away obj
• interpolate in between
⊕ no aliasing
⊕ efficient
weighted average for next px

Perspective interpolation
tex coord of vx → tex coord of px
• linear interp in screen space
wanted | we get
proj plane (non-linear)
Problem: non-linear variation in screen coords

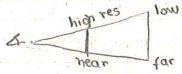
• Ambient → uniform independent of direction (general ambient color)
• Diffuse → general direction of light, which is reflected (radiosity) regardless the viewers position
• Specular (white) shiny light on small areas of obj.

• Light maps vs real time shading and shadow maps
⊖ take up storage ⊖ some lighting models don't work well with them
⊖ can't use for moving lights or objs
⊕ high quality shadows, lights ⊕ fast once precomputed ⊕ can use virtually as number of light sources

5.1a VISIBILITY & SHADOWS

Visibility problem: some parts of some surfaces are occluded
Painter's alg solution render objs/polygons from furthest to nearest
⊖ cyclic overlaps
⊖ intersections

Z-buffering store z-value ∀px, init to ∞
∀polygon: if z value smaller for one pixel, update it
⊖ limited res & non-linear resolution
→ set near plane far from cam



Shadows - depth cue
- scene lighting (es: point / area light)
- realism

1- planar shadows . draw projection of obj on ground
⊖ no self shadows / on other obj ⊖ curved surfaces not supported

2- projective texture shadows
- separate obstacle/receiver
- compute b/w image from light → use it as projective texture: ⊖ specify obs/receiver ⊖ no self shadows

Bias ?
avoid self-shadows → add bias (hard to choose)
$d(x_L) + bias < z_L$
es. too much / little

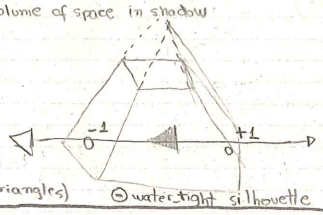
3- shadow maps (high end)
- compute depths from light camera
- ∀px on camera plane
  - find x in world coord
  - project on to light plane (x_L)
  - compare $d(x_L)$ shadow map & z_L (dist)
    $d(x_L) < z_L \Rightarrow x$ in shadow

⊖ FOV (field of view) es points outside light's FOV → use CUBICAL shadow map
⊖ aliasing (under sampling)
SOL • don't filter depth
• filter result of test → produces fake soft shadow

4- shadow volumes • explicitly represent volume of space in shadow
- if polygon inside → shadow
- similar to clipping
- naive O(#pol x #lights)

shoot ray from eye
each time boundary intersected → c++ / c--
if c > 0 → primitive in shadow
= 0 → in light

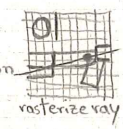
⊕ optimization: use silhouette edges only (back & front-facing meet of polygons)
⊖ a lot of geometry ⊖ expensive (skinny triangles) ⊖ watertight silhouette



Acceleration
• complex scene 600 years
cost = O (#rays x #obj) • acceleration: fewer intersection (11 Min)

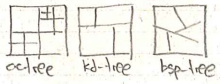
uniform grids
- compute BB
- set grid resolution
- store reference to raster-objects
- only check when possible intersection

⊕ fast to build
⊕ easy to code
⊖ non-adaptive to scene



traversal:
- incrementally rasterize ray
- step when intersection happens

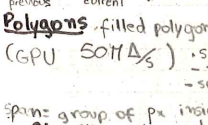
space partitioning trees



5.1b SCAN CONVERSION (rasterization)

Lines generate discrete px values

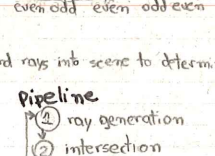
Bresenham line (choose closest px at each intersection) ⊕ fast decision of next px
Criterion: midpoint M
intersection q
line: $f(x,y) = ax + by + c = 0$ (implicit form)
$d = f(M) = f(x_p + 1, y_p + \frac{1}{2})$ >0 → NE, <0 → E, select



E	NE
$d_{old} = f(x_p+1, y_p+\frac{1}{2})$	$d_{old} =$ "
$d_{new} = f(x_p+2, y_p+\frac{1}{2})$	$d_{new} = f(x_p+2, y_p+\frac{3}{2})$
$d_{new} - d_{old} = a = \Delta y$	$d_{new} - d_{old} = a + b = \Delta y - \Delta x$

$a = \Delta y$
$b = -\Delta x$?

Polygons filled polygons, very important
(GPU SOME Δ's)
• spatial coherence
- naive: inside test ∀px inefficient
- scanline



span: group of px inside scanline
Alg: all scanline inters
- sort them by x
- fill spans if odd parity

5.2 RAY TRACING

• send rays into scene to determine px color

Forward

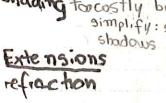
Backward w/ plane

Pipeline
1. ray generation
2. intersection
3. shading

Ray generation

super sampling → multiple rays per px

Intersection Ray $r(t) = \vec{o} + \vec{d}\,t$
sphere $\|x - c\|^2 - r^2 = 0$ (x = point of interest)
$\|o + td - c\| \dots$
triangle bary centric $x = s_1 P_1 + s_2 P_2 + s_3 P_3$
intersect plane $(o + td - p_1) \cdot n = 0 \rightarrow t = -\frac{(o - p_1) \cdot n}{d \cdot n}$
compute s_i (check $s_i \in [0,1]$)

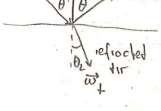


$n = (p_2 - p_1) \times (p_3 - p_1)$
$s_1 + s_2 + s_3 = 1$

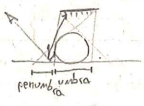
shading forcostly be physically accurate
simplify: surf reflectance (diff/spec/amb/transparency)
shadows (shadow rays for point in shadow/not)

Extensions

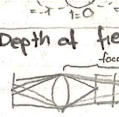
refraction



multiple lights/ DOF (soft shadows, multiple shadow rays)

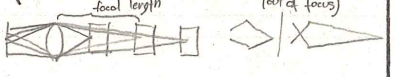


motion blur



sample obj & intersect in time

Depth of field (DOF)



7.1 ANIMATION

Animation → bringing images to life

what
- characters
- faces
- hair
- elastic materials
- natural phenomena
- cloth
- herds (fish)
- crowds

How
- keyframing
- skeletal (rig), character
- subspace param
- pbs, particle systems
- motion capture
- video based/facial
- example based (birds)
- procedural

Considerations
- control
- efficiency
- realism/style
- fast deformation with intuitive controls

controllers:
classical
sketch-based
keyframing

Animation pipeline
1) story-bording (planning)
2) concept design (sketches character/environm, features)
3) 3D modeling (geometry & textures)
4) rigging (embed animation controllers/skeleton/ready to keyframe)
5) blend-shapes creation (facial expression/generation using blending → linear space $(\sum w_i f_i)$ → face exp
6) animation (key-frames for controllers / interpolation)
7) post-effects (lighting/shading/render)

Rigging attach skeleton to model using bones (embed)
↳ keyframe to animate

Bone / Joint (hierarchy) rigid transformation

weights: indicate how much a vertex is influenced by bone

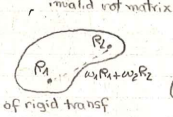
Rit;

T = transform (homogenous coords)

$T(x) = avg(T_1, T_2, w_1, w_2)$
$= w_1(x) T_1 + w_2(x) T_2$
$x' = T(x)x$ → rotation $w_1 R_1 + w_2 R_2$
transl $w_1 t_1 + w_2 t_2$ invalid rot matrix

valid: $R^T = R^{-1}$
$\det(R) = 1$

linear blend skinning
how to blend transf
⊖ problems: elbow



Manifold of rigid transf
(manif: topological space that resembles euclidean space)

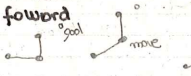
→ since $R \to q$ is not linear, also interpolation is not → interp. curve
using quat

Spherical blend skinning
blend using $(q_2 q_1^{-1})^{w_2} q_1$
for more quat, a good approx is $\sum w_i q_i$

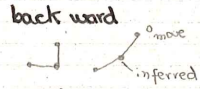
Rigid motion with dual quaternions (8 params - 2 constr = 6, like Rt) (blending)

kinematics

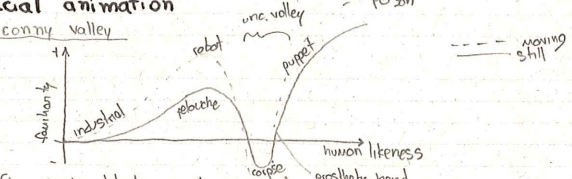
foward



back ward



Facial animation
uncanny valley

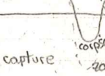


⚠ fine-scale details → capture

7.2 GEOMETRY PROCESSING

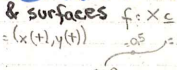
Parametric curves & surfaces $f: X \subseteq \mathbb{R}^M \to Y \subseteq \mathbb{R}^N$

curves 1D → 2D $s(t) = (x(t), y(t))$



circle $p(t) = r(\cos t, \sin t)$ $t \in [0, 2\pi[$

Bezier
$s(t) = \sum_{i=0}^{n} P_i B_i^n(t)$ $B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$

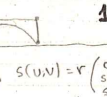


1D → 3D (curves in 3D)
$s(t) = (x(t), y(t), z(t))$

Surfaces 2D → 3D
$s(u,v) = (x(u,v), y, z)$

$s(u,v) = r \begin{pmatrix} \cos u \cos v \\ \sin u \cos v \\ \sin v \end{pmatrix}$ $(u,v) \in [0, 2\pi[\times [-\frac{\pi}{2}, \frac{\pi}{2}]$

Bezier
$s(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} B_i^m(u) B_j^n(v)$



normal: $s_u = \frac{\partial s(u,v)}{\partial u}$ $s_v = \frac{\partial s(u,v)}{\partial v}$ $n = (s_u \times s_v)$ norm

Volumetric repr 3D → 1D (color) (each point in space has color) MRI medic scan
⊕ easy to generate
⊕ point-wise properties
⊕ control by hand
⊖ hard determine in/on/out
⊖ reverse engineering

Implicit curves & surfaces $f: \mathbb{R}^M \to \mathbb{R}$ $S = \{x \in \mathbb{R}^2 \mid f(x) = 0\}$ $S = \{x \in \mathbb{R}^3 \mid \dots\}$

$f(x) \geq 0$ out
$= 0$ on
< 0 in

Surface normal
$\nabla f = (x,y,z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$

$f(x,y,z) = x^2 + y^2 + z^2 - r^2$

⊕ easy determine in/on/out
⊖ hard to generate points
⊕ easy to combine
⊖ limited set
⊖ no real time

Point set surfaces (scans) acquired data • point-wise attributes

local fitting approach

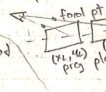
$f(x) = 0$

• implicit representation & fast proj
• find line for each neighborhood
⊕ robust to noise (Mesh conversion)
⊕ direct rendering
↳ in space, discretize with grid (dividing quad/tri)

⊕ easy in/on/out ⊕ easy generate points
⊕ reconstruction real data ⊕ real time ⊖ modeling

Shape acquisition (digitalization of real world)

1. 3D scanning (touch + optical) • active • passive (no control over light)



2. registration → all scans in same coord syst (different views)
iterative closest pt alg repeat (rigid motion) / feature based method (descriptors)

3. pre-processing (cleaning/repairing/resampling)
4. reconstruction → mathematical repres. of shape
VORONOI → connect points
APPROX → efficient, robust

Modeling tools / Editing geometry
interactive/sketch based UI
deformations (free/elastic/skeletal/struct aware)
cutting/fracturing
smoothing/filtering
compression/simplific/subdivision surfaces
differential properties/segmentation
symmetry/structure detection/feature extration
saliency

PBS: rigid body sim 6.2 CG

$p(t) = R(t)\,p_0 + x(t)$

$r_i(t)$ = position of particle (wrt com)? global

linear (velocity) $v = x'$ ← 2 Kinematics → $R' = \omega * R$ (pos change) (rot change)

R fixed

(angular velocity) col.wise × fixed (·x)

orientation

axis & speed ($\omega \in \mathbb{R}^3$)

$r(t) = o + b$

$r'(t) = \omega(t) \times b(t) = \omega \times r$

1 representation

$x \in \mathbb{R}^3$

$R \in \mathbb{R}^{3\times3}$

$com = \frac{\sum m_i r_i}{m}$

$R = [x'\ y'\ z']$

$F = \sum F_i$ (force)

$T = \sum T_i$ (torque)

$T_i = (r_i - x) \times F_i$

$q' = \frac{1}{2}(I^{-1}L)q$ (ω)

$F = ma = mv' = P'$

$P = mv$ (lin mom) [Ns]

$L = I\omega$ (angular mom)

↳ inertia tensor $\in \mathbb{R}^{3\times3}$ (mass distribution) relative to com

$P' = F$

$L' = T = Iw'$

$F \to P \to v \to x$

orthogonality: P indep R, ω; L indep x, v

→ * how to (pre)-compute it

inertia: momentum is conserved

$I_{body} = \sum_i m_i((r_{i,T} r_{i,}) 1 - r_{i,} r_{i,}^T)$

$I(t) = R \cdot I_{body} \cdot R^T$

state $S(t) = \begin{pmatrix} x \\ R \\ P \\ L \end{pmatrix}$ $S'(t) = \begin{pmatrix} v \\ \omega \cdot R \\ F \\ T \end{pmatrix}$

ODE: $x' = \dots$, $R' = \dots$

Simulation alg

precompute $x, \Pi^{-1}, I_{body}^{-1}$

compute: $F \leftarrow \sum F_i$, $T \leftarrow \sum (r_i - x) f_i$ ($F = \sum$, $T = \sum$)

update (euler): $x \leftarrow x + \Delta t\, \Pi^{-1} P$ ($x \mathrel{+}= \Delta t\, v = \Delta t\,(\Pi^{-1}P)$); $P \leftarrow P + \Delta t F$ ($P \mathrel{+}= \Delta t F$); $R \leftarrow R + \Delta t\,(\omega * R)$ ($R \mathrel{+}= \Delta t\, R' = \Delta t\,(\omega * R)$); $L \leftarrow L + \Delta t\, T$ ($L \mathrel{+}= \Delta t T$); $I^{-1} \leftarrow R\, I_{body}^{-1} R^T$ ($I^{-1} = R\, I_{body}^{-1} R^T$ (rot)); $\omega \leftarrow I^{-1} L$ ($\omega = I^{-1}L$)

per-particle quantities: $r_i \leftarrow x + R r_{i0}$ ($r_i = x + R\, r_{i0}$); $v_i \leftarrow v + \omega \times (r_i - x)$ ($v_i = v + \omega \times (r_i - x)$)

$N = \frac{M}{kg}$ $S2$

kin: how it moves

dyn: what causes it

Collision Handling

impulse $J = m\,\Delta v$, $\Delta P = J$, $J = j\hat{n}$ (j = scalar)

$J = \int F\,dt$ * complete

$\Delta L = T_{impulse} = (p - x(t)) \times J$] any mom

$v_{rel,aft} = -\varepsilon\, v_{rel,bef}$

stops ← $\varepsilon \in [0,1]$ → bouncy (restitution coefficient)

resting contact: $\forall$ contact p $\Rightarrow v_{rel} = 0$

$v_{rel} = \hat{n}\,(v_a - v_b) \gtrless 0$ (normal of contact pt)

impulse driven (manip.) | force driven (manip.)

$-\Delta v = 2\,(v \cdot n)\,n$

$v_f - v_i = \Delta v$

impulse: ⊕ fast sim, no timestep change, instant reaction; ⊖ collision sleeping

force: ⊕ easy, simply add forces, extends to multiple collisions in step; ⊖ slow sim, timestep adjusted (smaller), needs prepop time (interpenetration)

⊖ how to compute best to be correct, complex for multiple collisions (solve constrained prob)

q vs R:
⊕ more stable, compact (GPU eff), easier to interp, no gimbal lock ...
⊖ z axis parallel → loss of 1 deg of freedom

OPEN_GL

pos transl. to camera coord (8)

Code

VERTEX SHADER

(gl_Position = proj_mat * modelview_mat * position) (model → world → camera)

```
#version 440
                                  const over all vx
uniform mat4 projection;          // cam -> screen (view/cam proj matrix)
uniform  "   modelview;           // obj -> cam (rigid trans + world coord trans matrix)
uniform vec3 light_pos;           obj SPACE
in vec3 inPosition;               attributes: per vx data
in   "  inNormal;                // vx coord in model sp
out vec3 outPosition;             varying: vx shader output, input to rasterizer
   "     outNormal;
in vec3 color_in; //vx color

*
void main()
{
    outPosition = inPosition;                 color_out = C*(diffIntensity + ambientIntensity);
    outNormal = inNormal;                                  eg 0.3;

    gl_Position = projection * modelview * vec4(inPosition, 1.0f);
    gl_PointSize
    gl_ClipDistance
}
```

camera

FRAGMENT SHADER

```
#version 130
                                  varying
in  vec4 color_out;
in  float intensity;

void main(void)
{
    gl_FragColor = color_out;
}
```

TEXTURE

```
loadImage(&texture_data); // load & bind
glGenTextures(1,&tex id);
glBindTexture(GL_TEXTURE_2D, tex id);
glTexImage2D(  "          ,0,GL_RGB,...);
uniform sampler2D texUnit; //texUnit
varying vec2 texCoords;
    gl_FragColor = texture2D(texUnit, texCoords);
```

f(( ) || &&

```
* float computeDiffuseIntens(in vec3 pos, in vec3 normal, in vec3 lightPos) // compute intensity of reflected
{                                                                           diffuse light
    normal = normalize(normal);
    vec3 vecToLight = normalize(lightPos.xyz - position.xyz);
    float diffuseIntensity = dot(normal, vecToLight);
    diffuseIntensity = clamp(diffuseIntensity, 0.0, 1.0);
    return diffuse Intensity;
```

functions

```
c = mix(c1,c2,t);   // interpolate btw 2 colors, t∈[0,1]
    normalize
    dot
    clamp
    mod     trunc     fract
            int part  float part
    15.74 → 15        .74
```

MATLAB

```
A.*B                reshape(A,3,3)      sum(A,3)    conv2 // 2D conv      mean
Mask = A>T          A(find(A==2))=3                 fspecial('gaussian',hsize,6)   [PC,V] = eig(covariance)
A(:)                A(i,1)                          edge.                          Y = compressing - PC' * data
mov = aviread('bluescreen.avi');                    colormap('gray')
im = mov(frameNo).cdata;                            fftshift(fft2(g))
```

TOOLS

deg	rad	sin	cos	tan
0°	0	0	1	0
30°	π/6	1/2	√3/2	√3/3
45°	π/4	√2/2	√2/2	1
60°	π/3	√3/2	1/2	√3
90°	π/2	1	0	-

Quaternion rotation (↑)

7C. AUGMENTED REALITY

1. In air display

head mounted / head worn / head attached / handheld / spatial

HMD (see through) to tablets — Holodesk

2. tracking (user/cam)

- active (mech/magn/GPS/wifi/...)
- passive (accelerom/gyro) + inertial
- vision based (marker/natural features) tracking
- hybrid (combination)

markers: rect enough to estimate all

ARToolKit: search → find 3D pos → identify marker → position models → render

NFT (natural feat track) → descriptors — SIFT: create descriptors

3. scene reconstruction

localiz → camera pos est. — 6(5) DoF, use 2+ img (=tracking, X known, find T)

mapping → reconstruct env geom (T known, triang X)

$T = [R|t]$ camera

$\{x\}$ points in space

+ dealing with drift

SLAM (simult loc & mapping)

sparse map of high-qual features (mono-SLAM) — ICP

dense map of low-qual — Kinect voxel

4. user input

naive: push-pin particles

Rigid Body Physics

- Write ODE that govern dynamics of rigid body.
- $\dot{x}(t) = \frac{P(t)}{m}$ (mass, velocity)
- $\dot{P}(t) = F_i(t)$ — force applied to $r_i(t)$ — lin. momentum
- $\dot{q}(t) = \frac{1}{2}(I^{-1}(t)\,L(t))\,q(t)$ — rotation; I^{-1} moment of inertia tensor (known)
- $\dot{L}(t) = (r_i(t) - x(t)) \times F_i(t)$ — angular momentum

OF based particle sys

fg map → segment RGB img → group calc flow field → tracked patches → physics particles

Digits: freehand wrist-worn gloveless sensor

IK model: infer hand pos based on finger tips